IAS | INSTITUTE FOR APPLICATION SECURITY

# HyTrack

## Tracking You Across Apps and the Web Hydra-Style

Malte Wessels

June 13, 2025

# About me

- PhD Student @ Institute for Application Security / TU Braunschweig
- Interested in all things Security and Privacy
- CTF @ CyberTaskForce Zero
- Helping at datarequests.org / Datenanfragen.de e.V.

# Our Phones


Image: Adrien - Unsplash License

# Tracking

- Apps use tracking to profile users for personalized ads
- Or sell user data directly
- Often, developers don’t implement this themselves
  - They use libraries and SDKs
- Our model: tracking technique in a library

# Current Tracking Techniques

- Google AD ID
- Identifier for Advertisers (IDFA)

# Current Tracking Techniques

- Google AD ID
- Identifier for Advertisers (IDFA)




- 38400000-8CF0-11BD-B23E-10B96E40000D
- EA7583CD-A667-48BC-B806-42ECB2B48606
- 00000000-0000-0000-0000-000000000000
- 00000000-0000-0000-0000-000000000000

# Current Tracking Techniques

- Google AD ID
- Identifier for Advertisers (IDFA)


- 38400000-8CF0-11BD-B23E-10B96E40000D
- EA7583CD-A667-48BC-B806-42ECB2B48606
- 00000000-0000-0000-0000-000000000000
- 00000000-0000-0000-0000-000000000000

# Recently: *Local Mess*

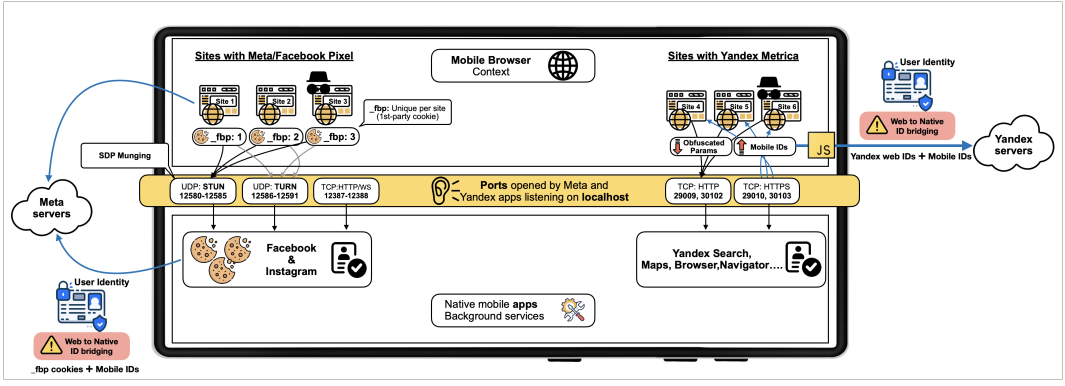



Image: Aniketh Girish, Gunes Acar, Narseo Vallina-Rodriguez, Nipuna Weerasekara, Tim Vlummens

We present a new tracking technique on Android  : HyTrack.

# Web Content on Android

- Four ways!
- Fire an `ACTION_VIEW` Intent (IPC Message)
- Opens the browser and loads the page [1/4]

# Web Content on Android

- Four ways!
- Fire an `ACTION_VIEW` Intent (IPC Message)
- Opens the browser and loads the page [1/4]

# WebView [2/4]

- Embed web content component.
- No batteries included.
  - Have to implement browser features yourself.
- History of security issues.

# WebView [2/4]

- Embed web content component.
- No batteries included.
  - Have to implement browser features yourself.
- History of security issues.

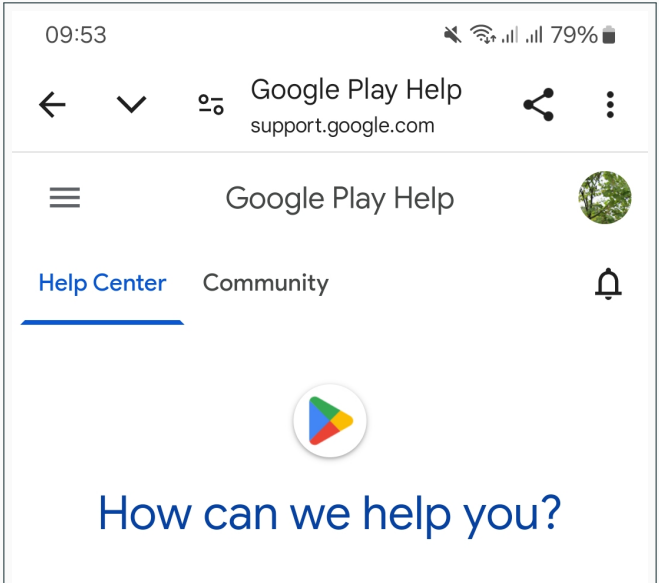
# Custom Tabs (CTs) [3/4]

- Open a special browser tab inside your app.
- Share the browser state.
- The look of the URL bar is customizable.
- Pros
  - Users are already logged in due to the shared state.
  - No context switch for users.
  - Devs do not have to re-implement browser features.

# Custom Tabs (CTs) [3/4]

- Open a special browser tab inside your app.
- Share the browser state.
- The look of the URL bar is customizable.
- Pros
  - Users are already logged in due to the shared state.
  - No context switch for users.
  - Devs do not have to re-implement browser features.

# Custom Tab Example [3/4]



# Trusted Web Activities [4/4]

- Custom Tabs can be upgraded to Trusted Web Activities (TWAs).
- TWAs are full-screen CTs.
  - Notice “Running in Chrome” depending on OEM, version, etc.
- They require a verified digital asset link between the app and the website.

# Trusted Web Activities [4/4]

- Custom Tabs can be upgraded to Trusted Web Activities (TWAs).
- TWAs are full-screen CTs.
  - Notice “Running in Chrome” depending on OEM, version, etc.
- They require a verified digital asset link between the app and the website.

# Digital Asset Links (DALs)

- Used to verify connection: app 🔗 website.
- Verified by the browser hosting the Trusted Web Activity.
- 🔗 Web: `.well-known` folder.
- 🔗 App: Defined in the app manifest.
- Website specified statically in the app or dynamically.

# (Static) DAL example

```
[{ // in the app
"relation": [
  "delegate_permission/common.handle_all_urls"
  ],
"target": {
  "namespace": "web",
  "site": "https://example.org"
}}]

[{ // on the web
"relation": [
  "delegate_permission/common.handle_all_urls"
  ],
"target" : {
  "namespace": "android_app",
  "package_name": "org.example.app",
  "sha256_cert_fingerprints": [
    "DE:CA:FB:AD:[...]:AA"
    ]
}}]
```

# (Static) DAL example

```
[{ // in the app
"relation": [
  "delegate_permission/common.handle_all_urls"
  ],
"target": {
  "namespace": "web",
  "site": "https://example.org"
}}]

[{ // on the web
"relation": [
  "delegate_permission/common.handle_all_urls"
  ],
"target" : {
  "namespace": "android_app",
  "package_name": "org.example.app",
  "sha256_cert_fingerprints": [
    "DE:CA:FB:AD:[...]:AA"
    ]
}}]
```

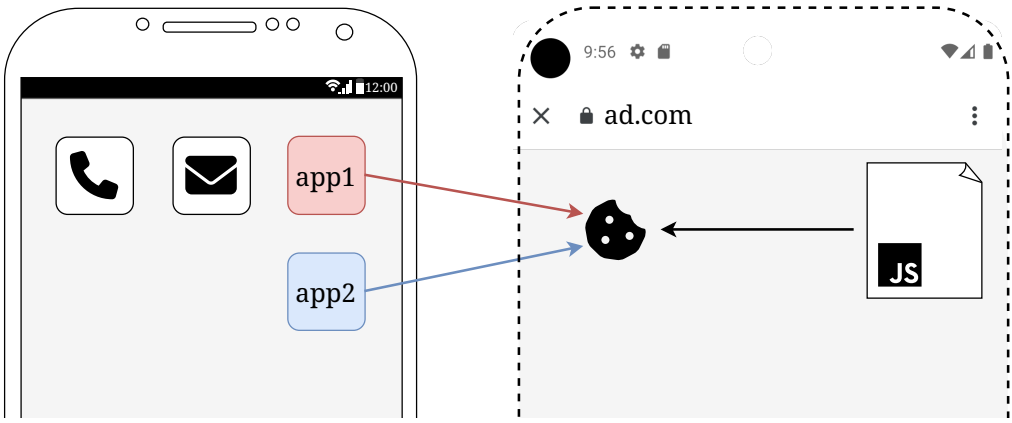
# HyTrack Idea

- Share ID via shared browser state of CTs and TWAs.
- Bring the power of web tracking to Android  .

# HyTrack Idea

- App A opens a Custom Tab to ad.com.
- ad.com sets a Cookie  containing the tracking ID.
- App B opens a Custom Tab to ad.com, the Cookie  is sent.
- Shared Browser State -> IDs are shared between CTs / Apps!

# Across Apps



# Across Apps and Web

- Websites can send tracking data to ad.com via web tracking techniques.
- This bridges the gap between native tracking and web tracking!

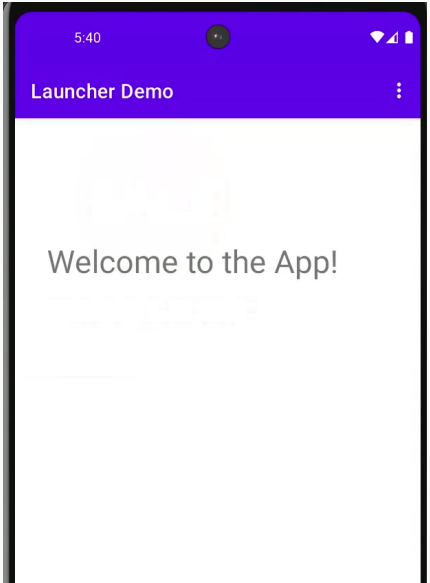
# Across Apps and Web

- Websites can send tracking data to ad.com via web tracking techniques.
- This bridges the gap between native tracking and web tracking!

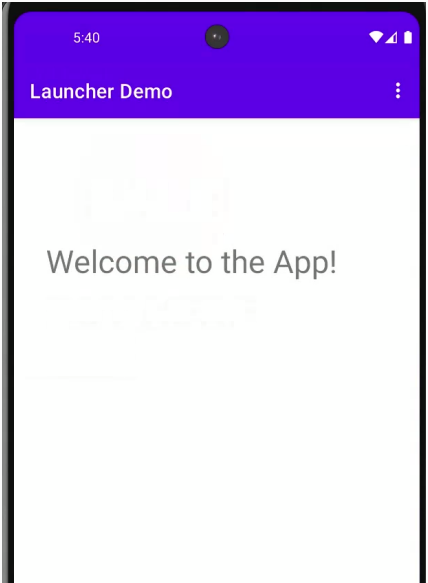
# Seamless Transition

- Trusted Web Activities are full-screen, so we can style the complete display.
- Hide transition, make it seamless.
  - Mimic the look of the native part of the application.
  - Easily done with cross-platform frameworks, e.g., material design.

# Screenshots



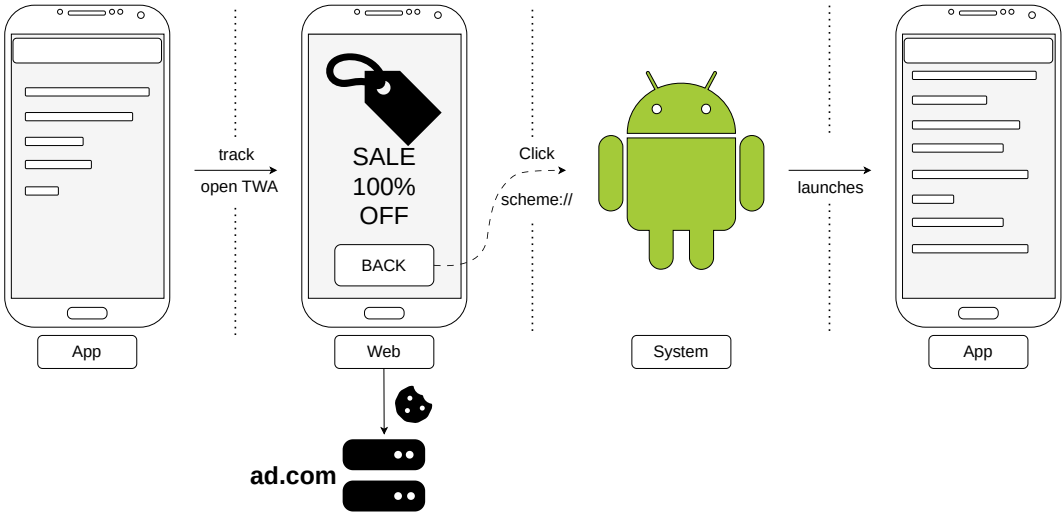
# Screenshots




# Navigation

- How to navigate back from the web content to the app?
- Custom URL Scheme!
- Apps can register a custom, unique URL scheme.
- Clicking on the scheme link will relaunch the app.

# Resulting UI/UX



# Hydra

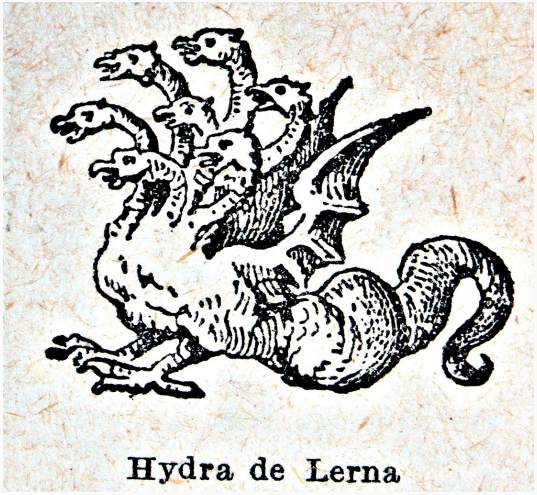
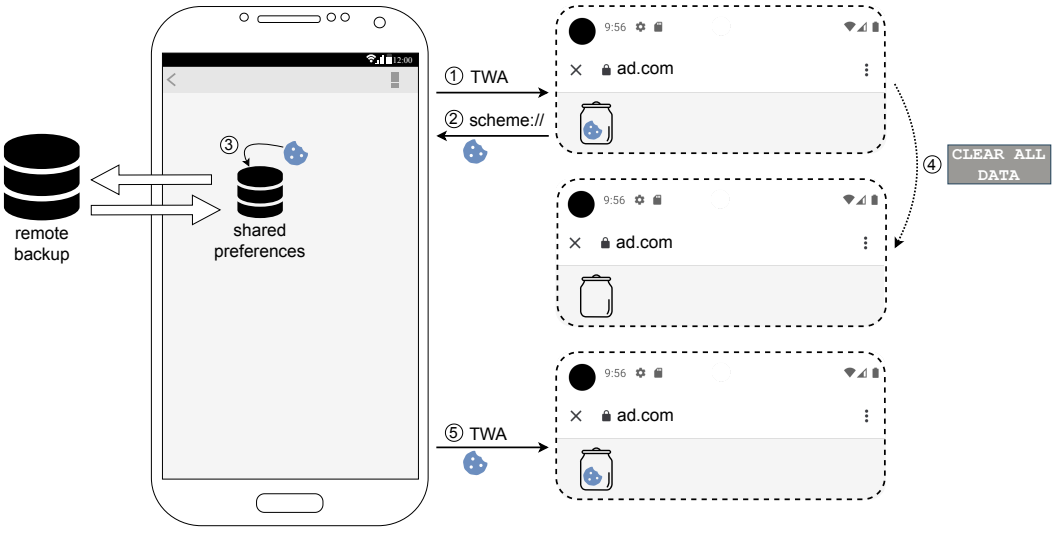



Image: Public Domain

# Resurrect & Persist



# How can we defeat HyTrack?

- If multiple apps use HyTrack, it's hard to get rid off.
- Clear browser storage, remove all apps, and reinstall them without backup.

# Studies

# Study 0: Am I affected?

- In the wild: Different Android flavours and browsers.
- Tested devices with their stock Android flavour and browser.
  - Google Pixel, Samsung Galaxy, Huawei, Xiaomi, Poco, Oneplus
  - GrapheneOS
- Various non-stock Browsers
  - Chrome, Opera, Firefox, Tor Browser, UC Browser, Brave

# Study 0: Am I affected?

- In the wild: Different Android flavours and browsers.
- Tested devices with their stock Android flavour and browser.
  - Google Pixel, Samsung Galaxy, Huawei, Xiaomi, Poco, Oneplus
  - GrapheneOS
- Various non-stock Browsers
  - Chrome, Opera, Firefox, Tor Browser, UC Browser, Brave

# Study 0: Am I affected?

- In the wild: Different Android flavours and browsers.
- Tested devices with their stock Android flavour and browser.
  - Google Pixel, Samsung Galaxy, Huawei, Xiaomi, Poco, Oneplus
  - GrapheneOS
- Various non-stock Browsers
  - Chrome, Opera, Firefox, Tor Browser, UC Browser, Brave

# Study 0: Device Configurations

- Every browser that supports CTs is affected!
- Chromium-based (CT and TWA)
  - Chrome, Vanadium, Brave, Samsung Internet
- And Firefox-based ones (only CT)
  - Firefox, Tor Browser
- The other browsers did not support Custom Tabs.

# Study 0: Device Configurations

- Every browser that supports CTs is affected!
- Chromium-based (CT and TWA)
  - Chrome, Vanadium, Brave, Samsung Internet
- And Firefox-based ones (only CT)
  - Firefox, Tor Browser
- The other browsers did not support Custom Tabs.

# App Data Set

- Pulled 4.4k apps from Google Play Store.
- Top 200 Lists by 32 categories in *June '24*.

# Study 1: Dynamic App Analyis

- No-touch traffic analysis with a real device.
- Start apps one by one and record incoming and outgoing traffic.
- Idea: Don’t reset the phone between apps; search for cookies across apps.

# Study 1: Setup

- Adapted framework from our previous work to record responses[1].
- Machine-in-the-Middle via `mitmproxy`
  - `https://mitmproxy.org/`
  - Google Pixel 6A with `root`, Lab Wifi, and Lab PC.
- Frida and `objection` for TLS interception and unpinning.
  - `https://github.com/sensepost/objection`

[1] `https://github.com/App-Analysis/scala-appanalyzer`

# Study 1: Dynamic App Analyis

- Average of 35.09 requests per app.
- Observed 45.7k cookies sent, and 10.5k received.
  - values shared across apps
  - same domain
  - CT/TWA user agent
  - manual removal of obvious non-tracking values
  - launches CT or TWA at start
- Inspection: No deployment of HyTrack but web-wrappers.

# Study 1: Dynamic App Analyis

- Additionally, we interacted with 95 apps for up to 15 minutes each.
- We checked the traffic, no findings.
- Dynamic analysis provides a lower bound.
  - Maybe not the right apps.
  - Maybe we did not discover the right functionality.
  - Measurement pipelines are tricky.

# Study 1: Dynamic App Analyis

- Additionally, we interacted with 95 apps for up to 15 minutes each.
- We checked the traffic, no findings.
- Dynamic analysis provides a lower bound.
  - Maybe not the right apps.
  - Maybe we did not discover the right functionality.
  - Measurement pipelines are tricky.

# Study 2: Static App Analyis

- Idea: Check if applications already use CTs and TWAs.
  - Search for calls to CT APIs and TWA APIs.
  - Check registered Custom URL schemes.
  - Search for DALs, both static and dynamic.

# Study 2: Static App Analyis

- Androguard is a reverse engineering tool
- Search for calls, constants, etc.
- We ignore code in Google libraries.
- Apps are obfuscated or minified.
- Static analysis provides an upper bound.

# Study 2: Static App Analyis

- 1.5k apps use Custom Tab APIs.
- 190 apps use Trusted Web Activity APIs.
- 244 include a DAL; 190 use dynamic DAL functions.
- 1.9k apps with unique scheme registered.
- Cluster of schemes
  - E.g. SSO: `fb123456`
- *Over 20 % of apps have the capabilities to deploy HyTrack already*!

# Study 3: Web Measurement

- Idea: Collect DALs and search for attack patterns.
- Three sources for domains:
  - General web (TRANCO 500 000)
  - Known web trackers (EasyPrivacy list)
  - Known mobile trackers (Exodus)
- For HyTrack, we'd expect multiple apps to link to the same website.
  - From different developers.
  - AndroZoo dataset to get Google Play developer metadata.

# Study 3: Web Measurement

- 564k different hosts in total.
- 14.7k valid DALs collected.
- Manually inspect relations with $\geq 3$ developers (92 total)
- All cases were benign.
  - E.g., branded banking apps that share one backend.

# And now?

# Mitigations

- HyTrack abuses the core idea of Custom Tabs!
- Custom Tabs are “Broken By Design”.
- Hard to mitigate without breaking Custom Tabs’ initial promise.
  - Limit Custom Tabs and Websites to one each.
  - Permission System (‘Do you want App X to use data from Website Y?’)
- We disclosed the issue to both Android’s issue tracker and the browser vendors.

Now is the perfect time to act!

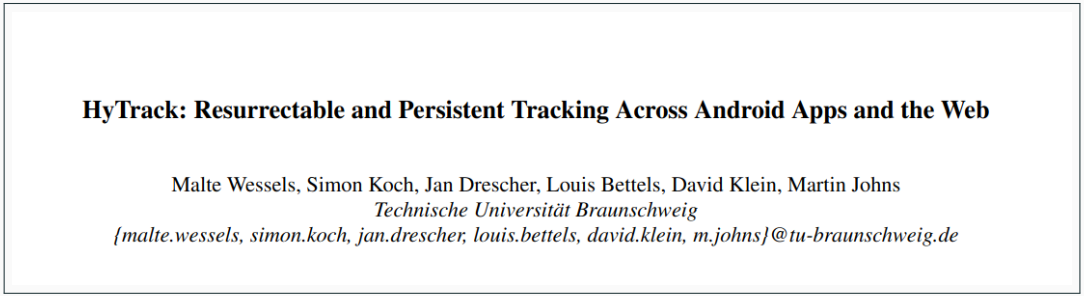
# End-User Mitigations

- Mitigation for users until the issue is fixed systematically
- Use a network-based traffic blocker
  - E.g., uBlock Origin in mobile Firefox
- Idea: When HyTrack is deployed, instances can be blocked.

# Conclusion

- We found HyTrack, a powerful tracking mechanism.
- Lots of apps have the pre-requirements!
- We should fix this!

# Paper



**HyTrack: Resurrectable and Persistent Tracking Across Android Apps and the Web**

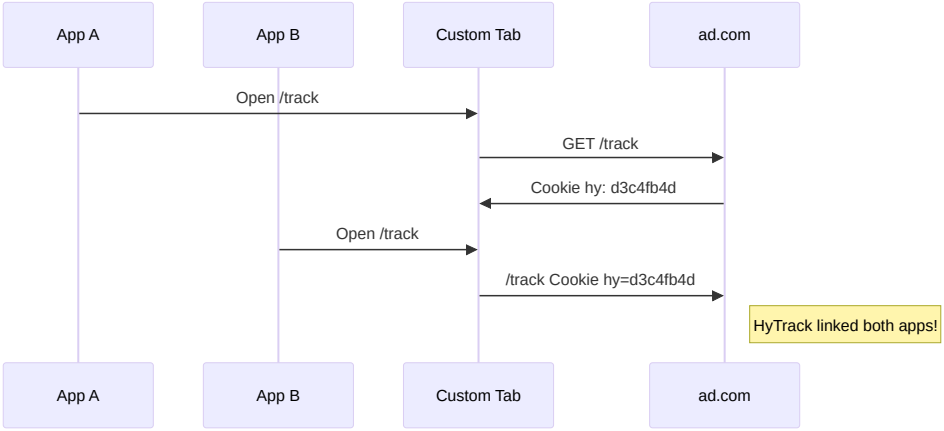
Malte Wessels, Simon Koch, Jan Drescher, Louis Bettels, David Klein, Martin Johns
*Technische Universität Braunschweig*
*{malte.wessels, simon.koch, jan.drescher, louis.bettels, david.klein, m.johns}@tu-braunschweig.de*

https://www.usenix.org/conference/usenixsecurity25/presentation/wessels

# Thank You!

- Any Questions?
- malte.wessels@tu-braunschweig.de
- maltee@chaos.social
- malte-wessels

# Example


App A
App B
Custom Tab
ad.com
Open /track
GET /track
Cookie hy: d3c4fb4d
Open /track
/track Cookie hy=d3c4fb4d
HyTrack linked both apps!
App A
App B
Custom Tab
ad.com